Distance_buffer.py

Function	Inputs	Output	Description
DistanceBuffer.__init__	max_size, min_variance_threshold	None	Initialize the DistanceBuffer.
DistanceBuffer.add_distance	distance, rssi	value	Add a new distance sample and optionally its corresponding RSSI.
DistanceBuffer.get_buffer	-	value	Return a shallow copy of the stored distance buffer.
DistanceBuffer.reset	-	None	Manually clear the buffer and stored RSSI history.
DistanceBuffer.is_full	-	value	Return True if the buffer is filled to max_size.

Distance_calculator.py

Function	Inputs	Output	Description
dbm_to_watt	dbm	value	Convert power from dBm to Watt.
watt_to_dbm	watt	value	Convert power from Watt to dBm.
channel_to_frequency_hz	channel	value	Map IEEE 802.11 Wi-Fi channel to center frequency in Hz.
wavelength_from_channel	channel	value	Return λ (meters) for a given Wi-Fi channel.
noise_floor_power	lambda_value, NF, T, index	value	Calculate noise floor in WATT using the full formula:
calculate_distance_from_rssi	RSSI_power, Pt, Gi, Gr, lambda_value, L, Pi, Pn_floor_power	value	Calculate distance 'd' using the EXACT formula:

Main_distance.py

Function	Inputs	Output	Description
dummy_client_interference_power	ap_id, client_mac, channel, timestamp	value	Placeholder for AP–client interference estimator.

Function	Inputs	Output	Description
dummy_ap_link_interference_power	tx_ap_id, rx_ap_id, channel, timestamp	value	Placeholder for AP–AP interference estimator.
update_telemetry_with_distance	telemetry_batch, get_interference_power	value	telemetry_batch: list[dict]
compute_ap_to_ap_rssi	telemetry_batch, ap_pair_distances, get_ap_interference_power	value	Compute RSSI between APs, assuming constant distances between APs.
_compute_single_ap_link	tx_ap, rx_ap, distance_m, channel, lambda_value, Temp, index, get_ap_interference_power, timestamp	value	Compute RSSI (in dBm) for a single AP->AP direction.
load_telemetry_json	filename	value	Load telemetry JSON from the same directory as this script.
save_telemetry_json	telemetry_batch, original_was_dict, original_path, out_filename	value	Save updated telemetry JSON alongside the original.
save_ap_ap_rssi_json	links, out_filename	value	Save AP–AP RSSI data to a separate JSON file.

Fast_loop_runtime.py

Function	Inputs	Output	Description
get_channel_interference_dbm	ap_telemetry, channel	value	Look up interference_power_dBm for a given 20 MHz channel from AP's channel_summary.
fallback_interference_by_counters	ap_telemetry	value	Fallback: if no channel_summary, decide interference by tx_failed/tx_retries.
channels_overlap_24ghz	ch1, ch2, bandwidth	value	True if two 2.4 GHz channels overlap at 20 MHz.
get_clients_count_on_channel	candidate_channel, telemetry_map	value	No description available.
score_channel	candidate_channel, ap_id,	value	Compute numeric score for candidate_channel on AP ap_id.

Function	Inputs	Output	Description
	telemetry_map, channel_occupancy, return_details		
choose_best_channel_for_ap	ap_id, telemetry_map, channel_occupancy, debug	value	Evaluate all candidate channels and return:
should_allow_dfs_as_last_resort	ap_telemetry, telemetry_map, channel_occupancy	value	Decide if we can use a DFS channel as last-resort:
classify_5ghz_40mhz_interference	ap_tel, thresh_dbm	value	For a 5 GHz, 40 MHz AP, classify interference as:
is_dfs_radar_like_event	ap_tel	value	DFS workaround:
generate_fastloop_proposals_from_snapshot	snapshot	list of dicts:	Use latest snapshot = telemetry_buffer.snapshot() and generate fast-loop proposals.

Rrm_controller_pretrain.py

Function	Inputs	Output	Description
visualize_rrm_graph_full	snapshot, channel_plan, step, out_dir	None	Full RRM graph for visualization / report:
fast_loop_worker	-	None	Background worker for interference-driven fast loop.
_lookup_channel_interference_power_from_summary	ap_id, channel, timestamp	value	Look up per-channel interference power for a given AP from its
interference_power	tx_ap_id, channel, timestamp	value	AP–AP interference power Pi (Watts) seen from tx_ap's perspective
client_interference_power	ap_id, client_mac, channel, timestamp	value	AP–client interference power Pi (Watts).
compute_guardrail_kpis	snapshot, changed_aps, fast_loop_stats	value	Compute KPI-style metrics for guardrails:
TelemetryBuffer.update	ap_id, telemetry	None	Merge new telemetry/summary into the existing record for this AP.

Function	Inputs	Output	Description
TelemetryBuffer.snapshot	-	value	Return a copy of all AP telemetry that is not older than max_age_sec.
update_snapshot_with_client_distances	snapshot	None	For each AP–client link in snapshot:
hash_mac	mac	value	Hash client MAC for privacy. Returns 8-char pseudonym.
classify_rssi	rssi	value	Classify client based on RSSI thresholds.
parse_rtt	text	value	Parse RTT summary text into a small dict like:
channel_freq_range_mhz	channel, width_mhz	value	Approximate Wi-Fi channel frequency range in MHz.
overlap_factor	ch_i, bw_i, ch_j, bw_j	value	Fractional spectral overlap in [0,1] between two (channel,width) pairs.
_compute_single_ap_to_ap_rssi	tx_ap, rx_ap, distance_m, lambda_value, Temp, index, Pi_combined_weighted, overlap	value	Compute RSSI at rx_ap due to tx_ap.
update_snapshot_with_ap_neighbors	snapshot	None	For AP pairs, compute AP–AP RSSI using Spectral Overlap logic.
build_networkx_interference_graph	snapshot	value	Nodes: AP IDs.
build_pyg_graph	snapshot, G	value	Node features per AP:
build_conflict_graph	snapshot, rssi_threshold	value	Builds the ‘Conflict Graph’ for DSATUR Channel Planning.
get_saturation_degree	G, node, coloring	value	Count unique colors assigned to neighbours.
dsatur_channel_plan	snapshot, prev_channel_plan, step_idx, dsatur_period	value	Build the CONFLICT graph, run sticky min-overlap DSATUR,
build_legal_actions_mask	snapshot, ap_index_map, now, min_change_interval	value	[num_nodes, NUM_ACTIONS] bool mask.

Function	Inputs	Output	Description
select_actions	q_net, data, legal_actions_mask, epsilon	value	Epsilon-greedy among legal actions.
_snap_bw	bw	value	Snap arbitrary bw to nearest discrete BW_VALUES.
_next_bw	current_bw, direction	value	Move to next bw in BW_VALUES in given direction (+1 or -1).
is_peak_hour	ts	value	Returns True if the given timestamp falls within the configured
compute_locality_allowed_aps	snapshot, G_rl	value	Select APs that are 'troubled' and should be eligible for RL changes:
apply_actions_via_mqtt	actions, snapshot, ap_index_map, channel_plan, min_change_interval, blast_radius_limit, allow_peak_changes, locality_allowed_aps, step_idx, phase	None	Combine channel_plan + RL actions → MQTT commands.
explain_decision	ap_id, idx, snapshot, G, q_vals, action	value	Generate a human-readable reason for the chosen action.
compute_step_reward	snapshot, changed_aps	value	Global reward for one slow-loop step.
detect_config_changes	prev_snapshot, curr_snapshot	value	Return list of AP IDs whose config changed between prev and current:
_log	event, path	None	RL-friendly controller/guardrail log.
log_experience	prev_snapshot, prev_actions, reward, curr_snapshot, step_idx, violated_guardrail, guardrail_reason, path, step_explanations	None	Append one transition to JSONL log:

Function	Inputs	Output	Description
snapshot_configs	snapshot	value	Store simple config (channel, bw, power, obss) per AP.
check_guardrails	now	value	Compare rolling KPI means (or medians) between baseline and RL windows.
visualize_interference_graph	G_rl, step_idx, output_dir	None	Visualizes the REAL interference (G_rl).

Rrm_controller_deploy.py

Function	Inputs	Output	Description
sim_time	-	value	Simulated 'wall clock' in seconds since epoch.
sim_time_str	-	value	Human-readable simulated time.
visualize_rrm_graph_full	snapshot, channel_plan, step, out_dir	None	Full RRM graph for visualization / report:
fast_loop_worker	-	None	Background worker for interference-driven fast loop.
_lookup_channel_interference_power_from_summary	ap_id, channel, timestamp	value	Look up per-channel interference power for a given AP from its
interference_power	tx_ap_id, channel, timestamp	value	AP–AP interference power Pi (Watts) seen from tx_ap's perspective
client_interference_power	ap_id, client_mac, channel, timestamp	value	AP–client interference power Pi (Watts).
compute_guardrail_kpis	snapshot, changed_aps, fast_loop_stats	value	Compute RL guardrail KPIs from current snapshot.
compute_reward_and_guardrails_for_step	kpis, baseline_history, step_idx	value	Turn KPIs into:
maybe_schedule_rollback	violated_guardrail, reward, last_good_reward	value	Decide whether to rollback to last_good_* snapshot.
TelemetryBuffer.__init__	max_age_sec	None	No description available.

Function	Inputs	Output	Description
TelemetryBuffer.update	ap_id, telemetry	None	Merge new telemetry/summary into the existing record for this AP.
TelemetryBuffer.snapshot	-	value	Return a copy of all AP telemetry that is not older than max_age_sec.
update_snapshot_with_client_distances	snapshot	None	For each AP–client link in snapshot:
hash_mac	mac	value	Hash client MAC for privacy. Returns 8-char pseudonym.
classify_rssi	rssi	value	Classify client based on RSSI thresholds.
parse_rtt	text	value	Parse RTT summary text into a small dict like:
channel_freq_range_mhz	channel, width_mhz	value	Approximate Wi-Fi channel frequency range in MHz.
overlap_factor	ch_i, bw_i, ch_j, bw_j	value	Fractional spectral overlap in [0,1] between two (channel,width) pairs.
_compute_single_ap_to_ap_rssi	tx_ap, rx_ap, distance_m, lambda_value, Temp, index, Pi_combined_weighted, overlap	value	Compute RSSI at rx_ap due to tx_ap.
update_snapshot_with_ap_neighbors	snapshot	None	For AP pairs, compute AP–AP RSSI using Spectral Overlap logic.
build_networkx_interference_graph	snapshot	value	Nodes: AP IDs.
build_pyg_graph	snapshot, G	value	Node features per AP:
build_conflict_graph	snapshot, rssi_threshold	value	Builds the ‘Conflict Graph’ for DSATUR Channel Planning.
get_saturation_degree	G, node, coloring	value	Count unique colors assigned to neighbours.
dsatur_channel_plan	snapshot, prev_channel_plan, step_idx, dsatur_period	value	Build the CONFLICT graph, run sticky min-overlap DSATUR,

Function	Inputs	Output	Description
build_legal_actions_mask	snapshot, ap_index_map, now, min_change_interval	value	[num_nodes, NUM_ACTIONS] bool mask.
select_actions	q_net, data, legal_actions_mask, epsilon	value	Epsilon-greedy among legal actions.
_snap_bw	bw	value	Snap arbitrary bw to nearest discrete BW_VALUES.
_next_bw	current_bw, direction	value	Move to next bw in BW_VALUES in given direction (+1 or -1).
can_apply_site_change	now	value	No description available.
is_peak_hour	ts	value	Returns True if the given timestamp falls within the configured
compute_locality_allowed_aps	snapshot, G_rl	value	Select APs that are 'troubled' and should be eligible for RL changes:
apply_actions_via_mqtt	actions, snapshot, ap_index_map, channel_plan, min_change_interval, blast_radius_limit, allow_peak_changes, locality_allowed_aps, step_idx, phase, rollback	None	Combine channel_plan + RL actions → MQTT commands.
explain_decision	ap_id, idx, snapshot, G, q_vals, action	value	Generate a human-readable reason for the chosen action.
compute_step_reward	snapshot, changed_aps	value	Global reward for one slow-loop step.
detect_config_changes	prev_snapshot, curr_snapshot	value	Return list of AP IDs whose config changed between prev and current:
_log	event, path	None	RL-friendly controller/guardrail log.

Function	Inputs	Output	Description
log_experience	prev_snapshot, prev_actions, reward, curr_snapshot, step_idx, violated_guardrail, guardrail_reason, path, step_explanations	None	Append one transition to JSONL log:
snapshot_configs	snapshot	value	Store simple config (channel, bw, power, obss) per AP.
apply_configs_via_mqtt	configs	None	No description available.
check_guardrails	now	value	Compare rolling KPI means (or medians) between baseline and RL windows.
visualize_ap_channel_coloring	snapshot, channel_plan, step, out_dir	None	Pretty, report-friendly graph coloring figure.
visualize_channel_graph	G_conflict, channel_plan, step_idx	None	Visualizes the DSATUR Plan (Conflict Graph).
visualize_interference_graph	G_rl, step_idx, output_dir	None	Visualizes the REAL interference (G_rl).

passive_rtt.py

Function	Input	Output	Description
print_final_stats	None	None	Prints final RTT statistics: median, P95, loss rate, variance.
parse_rtt	Scapy packet	None	Extracts TCP timestamps, computes RTT, stops after MAX_SAMPLES.

__main__ (sniffer)	None	None	Starts sniffing on IFACE and processes packets for RTT.

call_rtt.py

Function	Input	Output	Description
run_passive_rtt	None	text	Executes passive_rtt.py with timeout, returns stdout.
parse_passive_rtt_output	stdout	dict	Extracts samples, RTT stats, jitter, loss.
update_cusum	p95 (float)	tuple	Performs CUSUM spike detection.
write_json	metrics dict	None	Writes RTT + CUSUM metrics into a JSON file.
main_loop	None	None	Repeatedly runs passive_rtt, parses metrics, writes JSON.
__main__	None	None	Starts recurring monitoring loop.

API Documentation for Execute_rrm.py

Function Name	Definition / Purpose	Input Parameters	Output / Return
run(cmd)	Runs a shell command, prints it, and returns output with error handling.	cmd (str): Shell command.	str on success; None on failure.
safe_cmd(cmd)	Runs shell command, never raises; returns 'ERROR' on failure.	cmd (str): Shell command.	'ERROR' or decoded output.
get_current_band()	Parses 'iw' output to detect Wi-Fi band.	None	'2.4', '5', or 'unknown'.
obss_to_txpower(obss_pd)	Maps OBSS-PD threshold to TX power using linear model.	obss_pd (float): OBSS-PD value.	int: TX power (10–20 dBm).
apply_action(action, value)	Applies RRM action by editing config and restarting AP.	action (str); value (any).	None (side effects only).
on_message(c, u, msg)	MQTT callback parsing JSON actions and invoking apply_action.	c: client; u: userdata; msg: MQTT message.	None.
get_telemetry_raw()	Collects iw-based telemetry from AP.	None	dict: {iw_info, station_dump}.